SIAM J. COMPUT.
Vol. 38, No. 3, pp. 922–945

EFFICIENT ALGORITHMS FOR DESCRIPTION PROBLEMS OVER FINITE TOTALLY ORDERED DOMAINS*

ÁNGEL J. GIL†, MIKI HERMANN‡, GERNOT SALZER§, AND BRUNO ZANUTTINI¶

Dedicated to the memory of Peter Ružička (1947–2003)

Abstract. Given a finite set of vectors over a finite totally ordered domain, we study the problem of computing a constraint in conjunctive normal form such that the set of solutions for the produced constraint is identical to the original set. We develop an efficient polynomial-time algorithm for the general case, followed by specific polynomial-time algorithms producing Horn, dual Horn, and bijunctive formulas for sets of vectors closed under the operations of conjunction, disjunction, and median, respectively. Our results generalize the work of Dechter and Pearl on relational data, as well as the papers by Hébrard and Zanuttini. They complement the results of Hähnle et al. on multivalued logics and Jeavons et al. on the algebraic approach to constraints.

Key words. finite domain, description problems, algorithms, complexity

AMS subject classifications. 68Q25, 68T27, 68W40

DOI. 10.1137/050635900

1. Introduction and summary of results.

Constraint satisfaction problems constitute today a well-studied topic on the frontier of complexity, logic, combinatorics, and artificial intelligence. It is indeed well known that this framework allows us to encode many natural problems or knowledge bases. In principle, an instance of a constraint satisfaction problem is a finite set of variable vectors associated with an allowed set of values. A *model* is an assignment of values to all variables that satisfies every constraint. When a constraint satisfaction problem encodes a decision problem, the models represent its *solutions*. When it encodes some knowledge, the models represent possible combinations that the variables can assume in the described universe.

Constraints can be represented by means of a set of variable vectors associated with an allowed set of values. This representation is not always well suited; therefore, other representations have been introduced. The essence of the most studied alternative is the notion of a *relation*, making it easy to apply it within the database or knowledge base framework. We focus on the representation by formulas in conjunctive normal form with the literals taking the form $(x \leq d)$ and $(x \geq d)$, where x is a variable and d is an element from the given finite domain D, totally ordered by the relation $\leq$ (see Hähnle et al. [5, 6, 16]). We study in this paper the constraint *description* problem, i.e., the problem of converting a set of vectors M to a formula φ

*Received by the editors July 13, 2005; accepted for publication (in revised form) February 20, 2008; published electronically June 6, 2008. This work was supported by ÉGIDE 06606ZF, ÖAD Amadeus 18/2004, ANR CANAR ANR-06-BLAN-0383-02, and Acciónes Integradas Hispano-Austriacas HU2003-0043 and HU2005-0024. A preliminary extended abstract of this paper appeared as [13].

http://www.siam.org/journals/sicomp/38-3/63590.html

†Department of Economics, Universitat Pompeu Fabra, 08005 Barcelona, Spain (angel.gil@upf.edu).

‡LIX (UMR 7161), École Polytechnique, 91128 Palaiseau, France (hermann@lix.polytechnique.fr).

§Department of Computer Science, Technische Universität Wien, 1040 Wien, Austria (salzer@logic.at).

¶GREYC (UMR 6072), Université de Caen, 14032 Caen, France (zanutti@info.unicaen.fr).

in conjunctive normal form, such that its satisfying assignments $\mathrm{Sol}(\varphi)$ equals the original set M. We consider this problem first in its general setting without any restrictions imposed on the initial set of vectors. We continue by imposing several properties on the initial set, like the closure under the minimum, maximum, and median operations. We subsequently discover that these closure properties induce the description by Horn, dual Horn, and bijunctive constraints, respectively. Moreover, we give an elegant and unified solution to the *structure identification* problem of these three classes, as it was defined by Dechter and Pearl [11]. Given a set of vectors, this problem asks whether it can be represented by a formula of a special type, computing such a formula if the answer is affirmative.

The motivations to study description and identification problems are numerous. From the artificial intelligence point of view, description problems formalize the notion of exact acquisition of knowledge from examples. This means that they formalize situations where a system is given access to a set of examples and it is asked to compute a formula describing it exactly. Moreover, this representation takes usually less space than the original set of examples; thus it can be stored more easily in a knowledge base.

Satisfiability poses a keystone problem in artificial intelligence, automated deduction, databases, and verification. It is well known that the satisfiability problem for arbitrary constraints is an NP-complete problem. Therefore, it is important to look for restricted classes of constraints that admit polynomial algorithms deciding satisfiability. Horn, dual Horn, bijunctive, and affine constraints, in the Boolean case, constitute exactly these tractable classes, as was proved by Schaefer [24]. Thus the description problem for these four classes can be seen as storing specific knowledge into a knowledge base while we are required to respect its format. This problem is also known as *structure identification*, studied by Dechter with Pearl [11] and by Hébrard with Zanuttini [17, 25]. Another motivation for studying description problems comes from combinatorics. Indeed, since finding a solution for an instance of a constraint satisfaction problem is difficult in general but tractable in the four aforementioned cases, it is important to be able to recognize constraints belonging to these tractable cases.

The study of Boolean constraint satisfaction problems, especially their complexity questions, was started by Schaefer in [24], although he did not yet consider constraints explicitly. During the last ten years, constraints gained considerable interest in theoretical computer science. An excellent complexity classification of existing Boolean constraint satisfaction problems can be found in the monograph [10]. Jeavons et al. [9, 19, 20] started to study constraint satisfaction problems from an algebraic viewpoint. Feder, Kolaitis, and Vardi [12, 22] posed a general framework for the study of constraint satisfaction problems. A part of the research in constraint satisfaction problems requires the existence of efficient description and identification methods for special constraint classes.

Recently, there has been much progress on constraint satisfaction problems over domains with larger cardinality. Hell and Nešetřil [18] studied constraint satisfaction problems by means of graph homomorphisms. Bulatov [7] made a significant breakthrough with a generalization of Schaefer's result to three-element domains. He also proved a dichotomy theorem for conservative constraints over arbitrary domains [8]. On the other hand, Hähnle et al. [5, 6, 16] studied the complexity of satisfiability problems for many-valued logics that present yet another viewpoint of constraint satisfaction problems. We realized reading the previous articles on many-valued logics that in the presence of a total order the satisfiability problems for the Horn, dual

Horn, and bijunctive many-valued formulas of signed logic are decidable in polynomial time. We also noticed that Jeavons and Cooper [21] studied some aspects of tractable constraints on finite ordered domains from an algebraic standpoint. This led us to the idea to look more carefully at constraint description problems over finite totally ordered domains, developing a new formalism for constraints based on an already known concept of inequalities.

The purpose of our paper is manifold. We want to generalize the work of Dechter and Pearl [11], based on the more efficient algorithms for Boolean description problems by Hébrard and Zanuttini [17, 25]. We also want to complement the work of Hähnle et al. on many-valued logics. Finally, we want to provide a characterization by closure properties of polynomial-time decidable subcases of constraint satisfaction problems over finite totally ordered domains, which are straightforward generalizations of the known polynomial-time decidable Boolean cases.

2. Preliminaries. Let $D = \{0, \dots, n-1\}$ be a finite, totally ordered domain of cardinality n, and let V be a set of variables. For a variable $x \in V$ and a value $d \in D$, the inequalities $x \geq d$ and $x \leq d$ are called positive and negative *literal*, respectively. The set of *formulas* over D and V is inductively defined as follows:

- the logical constants *false* and *true* are formulas;
- literals are formulas;
- if φ and ψ are formulas, then the expressions $(\varphi \wedge \psi)$ and $(\varphi \vee \psi)$ are formulas.

We write $\varphi(x_1, \dots, x_\ell)$ to indicate that formula φ contains exactly the variables $x_1, \dots, x_\ell$. For convenience, we use the following shorthand notation:

- $x > d$ means $x \geq d+1$ for $d \in \{0, \dots, n-2\}$, and *false* otherwise;
- $x < d$ means $x \leq d-1$ for $d \in \{1, \dots, n-1\}$, and *false* otherwise;
- $x = d$ means $x \geq d \wedge x \leq d$;
- $\neg$*false* and $\neg$*true* mean *true* and *false*, respectively;
- $\neg(x \geq d)$, $\neg(x \leq d)$, $\neg(x > d)$, and $\neg(x < d)$ mean $x < d$, $x > d$, $x \leq d$, and $x \geq d$, respectively;
- $\neg(x = d)$ and $x \neq d$ both mean $x < d \vee x > d$;
- $\neg(\varphi \wedge \psi)$ and $\neg(\varphi \vee \psi)$ mean $\neg\varphi \vee \neg\psi$ and $\neg\varphi \wedge \neg\psi$, respectively.

Note that $x = d$ and $x \neq d$ asymptotically require the same space as their alternative notation, i.e., $O(\log n)$. Indeed, since d is bounded by n, its binary coding has length $O(\log n)$.

A *clause* is a disjunction of literals. It is a *Horn* clause if it contains at most one positive literal, *dual Horn* if it contains at most one negative literal, and *bijunctive* if it contains at most two literals. A formula is in *conjunctive normal form* (CNF) if it is a conjunction of clauses. It is a Horn, a dual Horn, or a bijunctive formula if it is a conjunction of Horn, dual Horn, or bijunctive clauses, respectively. Since the considered formulas in what follows are all in CNF, we will use the expression "formula" with a slight abuse of terminology also for CNF formulas, without explicitly specifying it.

Note that contrary to the Boolean case, a clause in our formalism can contain the same variable twice, in both a negative and a positive literal, without being reducible. However, such a clause can be assumed to contain not more than twice the same variable, since $x \geq d \vee x \geq d'$ can be reduced to $x \geq \min(d, d')$ and, dually, $x \leq d \vee x \leq d'$ can be reduced to $x \leq \max(d, d')$.

Example 2.1. Let $D = \{0, 1, 2, 3, 4\}$ be the domain for our running example. The expressions $x \leq 2$ and $y \geq 4$ are a negative and a positive literal, respectively. Instead of $x \leq 2$ and $y \geq 4$, we can also write $x < 3$ and $y > 3$, respectively. The disjunction

of literals $(x \le 2 \vee x \ge 4 \vee y \le 4)$ is a Horn clause, $(x \le 2 \vee x \ge 4 \vee y \ge 3)$ is a dual Horn clause, and $(x \le 2 \vee x \ge 4)$ is a bijunctive clause. The formula

$$\varphi(x,y) = (x \le 2 \vee x \ge 4 \vee y \le 4) \wedge (x \le 2 \vee x \ge 4 \vee y \ge 4) \wedge (x \le 2 \vee x \ge 3)$$

is in CNF. ∎

An *assignment* for a formula $\varphi(x_1, \ldots, x_\ell)$ is a mapping $m\colon \{x_1, \ldots, x_\ell\} \to D$ assigning a domain element $m(x)$ to each variable x. The *satisfaction relation* $m \models \varphi$ is inductively defined as follows:

- $m \models \mathit{true}$ and $m \not\models \mathit{false}$;
- $m \models x \le d$ if $m(x) \le d$, and $m \models x \ge d$ if $m(x) \ge d$;
- $m \models \varphi \wedge \psi$ if $m \models \varphi$ and $m \models \psi$;
- $m \models \varphi \vee \psi$ if $m \models \varphi$ or $m \models \psi$.

The set of all assignments satisfying φ is denoted by $\mathrm{Sol}(\varphi)$, also called *models* of φ. If we arrange the variables in some arbitrary but fixed order, say, as a vector $x = (x_1, \ldots, x_\ell)$, then the models can be identified with the vectors in D^ℓ. The jth component of a vector m, denoted by $m[j]$, gives the value of the jth variable, i.e., $m(x_j) = m[j]$. The operations of conjunction, disjunction, addition, and median on vectors $m, m', m'' \in D^\ell$ are defined as follows:

$$\begin{aligned} m \wedge m' &= (\min(m[1], m'[1]), \ldots, \min(m[\ell], m'[\ell])), \\ m \vee m' &= (\max(m[1], m'[1]), \ldots, \max(m[\ell], m'[\ell])), \\ \mathrm{med}(m, m', m'') &= (\mathrm{med}(m[1], m'[1], m''[1]), \ldots, \mathrm{med}(m[\ell], m'[\ell], m''[\ell])). \end{aligned}$$

The ternary *median* operator is defined as follows: for each choice of three values $a, b, c \in D$ such that $a \le b \le c$, we have $\mathrm{med}(a,b,c) = b$. Moreover, median is a permutative operator; i.e., the identity $\mathrm{med}(a,b,c) = \mathrm{med}(\pi(a), \pi(b), \pi(c))$ holds for every permutation π on all domain elements $a, b, c \in D$. Note that the median can also be defined by $\mathrm{med}(a,b,c) = \min(\max(a,b), \max(b,c), \max(c,a))$ as well as by $\mathrm{med}(a,b,c) = \max(\min(a,b), \min(b,c), \min(c,a))$, which implies the identities

$$\begin{aligned} \mathrm{med}(m_1, m_2, m_3) &= (m_1 \vee m_2) \wedge (m_2 \vee m_3) \wedge (m_3 \vee m_1) \\ &= (m_1 \wedge m_2) \vee (m_2 \wedge m_3) \vee (m_3 \wedge m_1) \end{aligned}$$

for all vectors $m_1, m_2, m_3 \in D^\ell$.

Example 2.2. Consider the set of vectors $M = \{010, 013, 220, 440, 444\}$. It is closed under conjunction, since for each pair of vectors $m, m' \in M$ we have $m \wedge m' \in M$. It is *not* closed under disjunction, since $013 \vee 220 = 223 \notin M$. It is also *not* closed under median, since $\mathrm{med}(013, 220, 444) = 223 \notin M$. ∎

3. Formulas in conjunctive normal form. We investigate first the description problem for arbitrary sets of vectors.

Problem: DESCRIPTION.
Input: A finite set of vectors $M \subseteq D^\ell$ over a finite totally ordered domain D.
Output: A formula $\varphi(x_1, \ldots, x_\ell)$ over D in CNF such that $\mathrm{Sol}(\varphi) = M$.

The naive approach to this problem is to compute first the complement set $\bar{M} = D^\ell \setminus M$, followed by the construction of a clause $c(\bar{m})$ for each vector $\bar{m} \in \bar{M}$ missing from M such that $\bar{m}$ is the unique vector falsifying $c(\bar{m})$. The formula φ is then the conjunction of the clauses $c(\bar{m})$ for all missing vectors $\bar{m} \in \bar{M}$. However, this algorithm is essentially exponential, since the complement set $\bar{M}$ can be exponentially bigger than the original set of vectors M.

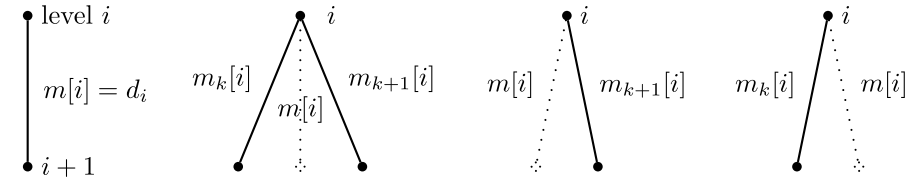
Example 3.1. Consider again the set of vectors $M = \{010, 013, 220, 440, 444\}$ as in Example 2.2. The complement set $\bar{M}$ contains $5^3 - 5 = 120$ vectors, where the lexicographically first ones are $000, 001, \ldots, 004, 011, 012$ and the last ones are $434, 441, 442, 443$. For example, the clause $c(001)$ is $(x_1 \neq 0 \vee x_2 \neq 0 \vee x_3 \neq 1)$, equivalent to $(x_1 \geq 1 \vee x_2 \geq 1 \vee x_3 \leq 0 \vee x_3 \geq 2)$. The only assignment to the variables x_1, x_2, x_3 which does not satisfy the clause $c(001)$ is the vector 001. Similarly, the clause $c(223)$ is $(x_1 \leq 1 \vee x_1 \geq 3 \vee x_2 \leq 1 \vee x_2 \geq 3 \vee x_3 \leq 2 \vee x_3 \geq 4)$. The formula φ describing M and built in this manner contains exactly 120 clauses. ∎

We present a new algorithm running in polynomial time and producing a CNF formula of polynomial length with respect to the cardinality of the set M, the dimension of vectors ℓ, and the size $O(\log |D|)$ of the domain elements in binary notation.

In what follows we assume without loss of generality the set of vectors M to be nonempty, which simplifies the presentation. Note, however, that the empty set $\emptyset$ is easily recognized and described by the formula $(x_1 \leq 0) \wedge (x_1 \geq 1)$, which is logically equivalent to *false*.

To construct the formula φ we proceed in the following way. We arrange the set M into an ordered n-ary semantic tree T_M [14], with branches corresponding to the vectors in M. In case M contains all possible vectors, i.e., $M = D^\ell$, T_M is a complete tree of branching factor $|D|$ and depth ℓ. Otherwise, some branches are missing, leading to gaps in the tree. We characterize these gaps by conjunctions of literals. Their disjunction yields a complete description of all vectors that are *missing* from M. Finally, by negation and de Morgan's laws we obtain φ.

Let T_M be an ordered tree with edges labeled by domain elements such that each path from the root to a leaf corresponds to a vector in M. The tree T_M contains a path labeled $d_1.\cdots.d_i$ from the root to some node if there is a vector $m \in M$ such that $m[j] = d_j$ holds for every $j = 1, \ldots, i$. The level of a node is its distance to the root plus 1; i.e., the root is at level 1 and a node reachable via $d_1.\cdots.d_i$ is at level $i + 1$ (Figure 3.1(a)). Note that all leaves are at level $\ell + 1$. If the edges between a node and its children are sorted in ascending order according to their labels, then traversing the leaves from left to right enumerates the vectors of M in lexicographic order, say, $m_1, \ldots, m_{|M|}$. A vector m is lexicographically smaller than a vector m' if there is a level i such that $m[i] < m'[i]$ holds, and for all $j < i$ we have $m[j] = m'[j]$.



(a) Level i (b) Missing between (c) ... left (d) ... right

FIG. 3.1. *Tree representation of vectors.*

Example 3.2. Let $M = \{m_1 = 010, m_2 = 013, m_3 = 220, m_4 = 440, m_5 = 444\}$ be the set of vectors over the domain $D = \{0, 1, 2, 3, 4\}$ for which we want to construct a formula φ in CNF satisfying the condition $\mathrm{Sol}(\varphi) = M$. The tree T_M is depicted in Figure 3.2 in solid lines. ∎

Suppose that m_k and m_{k+1} are immediate neighbors in the lexicographic enumeration of M, and let m be a vector lexicographically in between, thus missing from M. There are three possibilities for the path corresponding to m. It may leave the tree

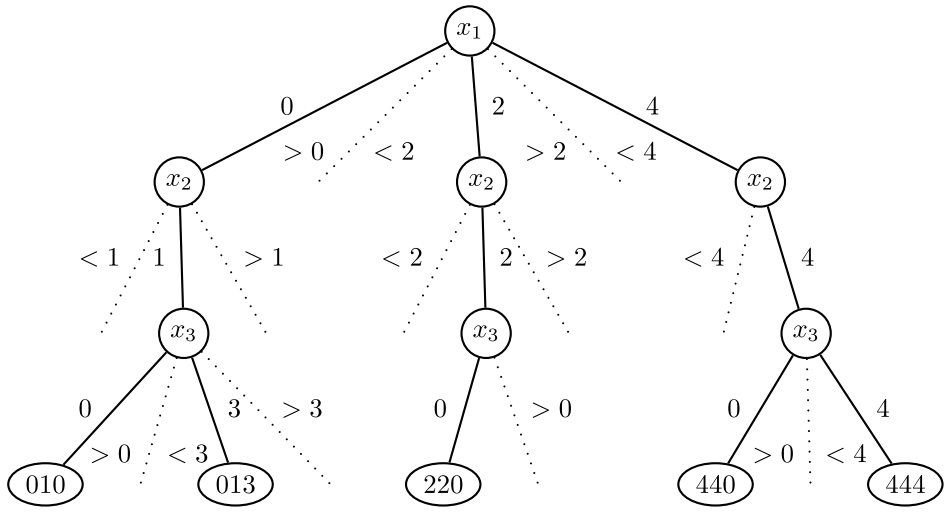


FIG. 3.2. *Tree T_M and the missing parts for the set $M = \{010, 013, 220, 440, 444\}$.*

at the fork *between* m_k and m_{k+1} (Figure 3.1(b)), or at the fork to the *left* of m_{k+1} (Figure 3.1(c)), or at the fork to the *right* of m_k (Figure 3.1(d)). Let i be the least position in which the consecutive vectors m_k and m_{k+1} differ. In other words, we have $m_k[i] \neq m_{k+1}[i]$ and $m_k[j] = m_{k+1}[j]$ for all $j < i$. The set of missing vectors can be characterized by the conjunctions middle(k, i), left$(k+1, i)$, and right(k, i), defined as follows:

$$\begin{aligned}
\text{middle}(k,i) &= \bigwedge_{j<i}(x_j = m_k[j]) \quad \wedge \quad (x_i > m_k[i]) \ \wedge \ (x_i < m_{k+1}[i]),\\
\text{left}(k+1,i) &= \bigwedge_{j<i}(x_j = m_{k+1}[j]) \quad \wedge \quad (x_i < m_{k+1}[i]),\\
\text{right}(k,i) &= \bigwedge_{j<i}(x_j = m_k[j]) \quad \wedge \quad (x_i > m_k[i]).
\end{aligned}$$

The situation depicted in Figure 3.1 is a snapshot at level i of the tree T_M.

Example 3.3. The missing parts in the tree T_M, displayed in Figure 3.2 by dotted lines, are described by the following conjunctions. First are the parts missing between two vectors,

$$\begin{aligned}
\text{middle}(1,3) &= (x_1 = 0) \wedge (x_2 = 1) \wedge (x_3 > 0) \wedge (x_3 < 3),\\
\text{middle}(2,1) &= (x_1 > 0) \wedge (x_1 < 2),\\
\text{middle}(3,1) &= (x_1 > 2) \wedge (x_1 < 4),\\
\text{middle}(4,3) &= (x_1 = 4) \wedge (x_2 = 4) \wedge (x_3 > 0) \wedge (x_3 < 4),
\end{aligned}$$

followed by the parts missing to the left,

$$\begin{aligned}
\text{left}(1,2) &= (x_1 = 0) \wedge (x_2 < 1),\\
\text{left}(3,2) &= (x_1 = 2) \wedge (x_2 < 2),\\
\text{left}(4,2) &= (x_1 = 4) \wedge (x_2 < 4),
\end{aligned}$$

and finally the parts missing to the right,

$$\begin{aligned}
\text{right}(2,2) &= (x_1 = 0) \wedge (x_2 > 1),\\
\text{right}(2,3) &= (x_1 = 0) \wedge (x_2 = 1) \wedge (x_3 > 3),\\
\text{right}(3,2) &= (x_1 = 2) \wedge (x_2 > 2),\\
\text{right}(3,3) &= (x_1 = 2) \wedge (x_2 = 2) \wedge (x_3 > 0).
\end{aligned}$$

There are no other missing parts in the tree T_M. ∎

To describe *all* vectors missing from M we form the disjunction of the above conjunctions for appropriate values of k and i. We need to determine the levels at which neighboring models fork by means of the following function:

$$\text{fork}(k) \quad = \quad \begin{cases} 0 & \text{for } k = 0,\\ \min\{i \mid m_k[i] \neq m_{k+1}[i]\} & \text{for } k = 1, \ldots, |M| - 1,\\ 0 & \text{for } k = |M|. \end{cases}$$

The values fork(0) and fork($|M|$) correspond to imaginary models m_0 and $m_{|M|+1}$ forking at a level above the root. They allow us to write the conditions below in a concise way at the left and right borders of the tree. The three situations in Figure 3.1 can now be specified by the following conditions:

$$\begin{array}{lcll}
i = \text{fork}(k) & \wedge & m_k[i] + 1 < m_{k+1}[i] & \text{(edges missing in between)},\\
\text{fork}(k) < i & \wedge & m_{k+1}[i] > 0 & (\ldots \text{ to the left}),\\
\text{fork}(k) < i & \wedge & m_k[i] < |D| - 1 & (\ldots \text{ to the right}).
\end{array}$$

The second condition in each line ensures that there is at least one missing edge. It avoids the conjunctions middle(k,i), left$(k+1,i)$, and right(k,i) to evaluate to false.

Example 3.4. The function fork for $M = \{m_1 = 010,\ m_2 = 013,\ m_3 = 220,\ m_4 = 440,\ m_5 = 444\}$ is given by the following table:

	0	1	2	3	4	5
fork	0	3	1	1	3	0

The aforementioned fork conditions for missing edges are satisfied in the following cases.

(i) For the first condition, implying edges missing in between, we have four cases where it is satisfied:

$$\begin{array}{lcl}
3 = \text{fork}(1) & \wedge & 1 = m_1[3] + 1 < m_2[3] = 3,\\
1 = \text{fork}(2) & \wedge & 1 = m_2[1] + 1 < m_3[1] = 2,\\
1 = \text{fork}(3) & \wedge & 3 = m_3[1] + 1 < m_4[1] = 4,\\
3 = \text{fork}(4) & \wedge & 1 = m_4[3] + 1 < m_5[3] = 4.
\end{array}$$

(ii) For the second condition, implying edges missing to the left, we have three cases where it is satisfied:

$$\begin{array}{lcl}
\text{fork}(0) < 2 & \wedge & 1 = m_1[2] > 0,\\
\text{fork}(2) < 2 & \wedge & 2 = m_3[2] > 0,\\
\text{fork}(3) < 2 & \wedge & 4 = m_4[2] > 0.
\end{array}$$

(iii) Finally, for the third condition, implying edges missing to the right, we have also three cases where it is satisfied:

$$\begin{aligned}
&\mathrm{fork}(2) < 2 \quad \wedge \quad 1 = m_2[2] < 4,\\
&\mathrm{fork}(2) < 3 \quad \wedge \quad 3 = m_2[3] < 4,\\
&\mathrm{fork}(3) < 2 \quad \wedge \quad 2 = m_3[2] < 4,\\
&\mathrm{fork}(3) < 3 \quad \wedge \quad 0 = m_3[3] < 4.
\end{aligned}$$

It can be easily seen that these conditions trigger the middle, left, and right formulas shown in Example 3.3. □

The disjunction of terms $\mathrm{middle}(k,i)$, $\mathrm{left}(k+1,i)$, and $\mathrm{right}(k,i)$ that satisfy the first, second, and third condition, respectively, for all models and all levels represents a disjunctive formula satisfied by the models *missing* from M. After applying negation and de Morgan's laws, we arrive at the required formula in CNF,

$$\begin{aligned}
\varphi(M) \quad &= \quad \bigwedge\{\, \neg\,\mathrm{middle}(k,i) \mid 0 < k < |M|,\ i = \mathrm{fork}(k),\ m_k[i] + 1 < m_{k+1}[i] \,\}\\
&\wedge \quad \bigwedge\{\, \neg\,\mathrm{left}(k+1,i) \mid 0 \le k < |M|,\ \mathrm{fork}(k) < i \le \ell,\ m_{k+1}[i] > 0 \,\}\\
&\wedge \quad \bigwedge\{\, \neg\,\mathrm{right}(k,i) \quad \mid 0 < k \le |M|,\ \mathrm{fork}(k) < i \le \ell,\ m_k[i] < |D| - 1 \,\},
\end{aligned}$$

where the condition $\mathrm{Sol}(\varphi) = M$ holds. Note that we use the negation symbol not as an operator on the syntax level but as a metanotation expressing that the formula following the negation sign has to be replaced by its dual. Note also that the conjunct $\mathrm{left}(k+1,i)$ is defined and used with the shifted parameter $k+1$ since it characterizes a gap lexicographically before the vector m_{k+1}.

Example 3.5. The set of vectors $M = \{010, 013, 220, 440, 444\}$ is described by the following CNF formula:

$$\begin{aligned}
\varphi(M) \ = \ &\neg\,\mathrm{middle}(1,3) \ \wedge\ \neg\,\mathrm{middle}(2,1) \ \wedge\ \neg\,\mathrm{middle}(3,1) \ \wedge\ \mathrm{middle}(4,3)\\
&\wedge\ \neg\,\mathrm{left}(1,2) \ \wedge\ \neg\,\mathrm{left}(3,2) \ \wedge\ \neg\,\mathrm{left}(4,2)\\
&\wedge\ \neg\,\mathrm{right}(2,2) \ \wedge\ \neg\,\mathrm{right}(2,3) \ \wedge\ \neg\,\mathrm{right}(3,2) \ \wedge\ \neg\,\mathrm{right}(3,3).
\end{aligned}$$

After substitution and application of de Morgan laws, this amounts to

$$\begin{aligned}
\varphi(M) \ = \ &(x_1 \neq 0 \vee x_2 \neq 1 \vee x_3 \le 0 \vee x_3 \ge 3) \ \wedge\ (x_1 \le 0 \vee x_1 \ge 2)\\
&\wedge\ (x_1 \le 2 \vee x_1 \ge 4) \ \wedge\ (x_1 \neq 4 \vee x_2 \neq 4 \vee x_3 \le 0 \vee x_3 \ge 4)\\
&\wedge\ (x_1 \neq 0 \vee x_2 \ge 1) \ \wedge\ (x_1 \neq 2 \vee x_2 \ge 2) \ \wedge\ (x_1 \neq 4 \vee x_2 \ge 4)\\
&\wedge\ (x_1 \neq 0 \vee x_2 \le 1) \ \wedge\ (x_1 \neq 0 \vee x_2 \neq 1 \vee x_3 \le 3) \ \wedge\ (x_1 \neq 2 \vee x_2 \le 2)\\
&\wedge\ (x_1 \neq 2 \vee x_2 \neq 2 \vee x_3 \le 0).
\end{aligned}$$

Replacing the shorthand notation $\neq$ by proper literals gives the following final formula:

$$\begin{aligned}
\varphi(M) \ = \ &(x_1 \ge 1 \vee x_2 \le 0 \vee x_2 \ge 2 \vee x_3 \le 0 \vee x_3 \ge 3) \ \wedge\ (x_1 \le 0 \vee x_1 \ge 2)\\
&\wedge\ (x_1 \le 2 \vee x_1 \ge 4) \ \wedge\ (x_1 \le 3 \vee x_2 \le 3 \vee x_3 \le 0 \vee x_3 \ge 4)\\
&\wedge\ (x_1 \ge 1 \vee x_2 \ge 1) \ \wedge\ (x_1 \le 1 \vee x_1 \ge 3 \vee x_2 \ge 2) \ \wedge\ (x_1 \le 3 \vee x_2 \ge 4)\\
&\wedge\ (x_1 \ge 1 \vee x_2 \le 1) \ \wedge\ (x_1 \ge 1 \vee x_2 \le 0 \vee x_2 \ge 2 \vee x_3 \le 3)\\
&\wedge\ (x_1 \le 1 \vee x_1 \ge 3 \vee x_2 \le 2) \wedge (x_1 \le 1 \vee x_1 \ge 3 \vee x_2 \le 1 \vee x_2 \ge 3 \vee x_3 \le 0).
\end{aligned}$$

Algorithm: DESCRIPTION
Input: Nonempty set $M \subseteq D^\ell$ of vectors.
Output: Formula $\varphi(M)$ in CNF, satisfying $\mathrm{Sol}(\varphi) = M$.
Method:

```
1: let M = (m_1, ..., m_|M|) be lexicographically sorted
2: φ(M) ← true
3: fork ← FORK(M)
4: for k ← 0 to |M| do
5:     f ← fork[k]
6:     if f > 0 and m_k[f] + 1 < m_{k+1}[f] then
7:         φ(M) ← φ(M) ∧ ¬middle(k, f)
8:     end if
9:     for i ← f + 1 to ℓ do
10:        if k < |M| and m_{k+1}[i] > 0 then
11:            φ(M) ← φ(M) ∧ ¬left(k + 1, i)
12:        end if
13:        if k > 0 and m_k[i] < |D| − 1 then
14:            φ(M) ← φ(M) ∧ ¬right(k, i)
15:        end if
16:    end for
17: end for
18: return φ(M)
```

Algorithm: FORK
Input: Lexicographically sorted nonempty list $M \subseteq D^\ell$ of vectors without duplicates.
Output: Array fork: $[0 \dots |M|]$ containing the fork function for M.
Method:

```
1: fork[0] ← 0
2: fork[|M|] ← 0
3: for k ← 1 to |M| − 1 do
4:     i ← 1
5:     while m_k[i] = m_{k+1}[i] do
6:         i ← i + 1
7:     end while
8:     fork[k] ← i
9: end for
10: return fork
```

FIG. 3.3. *Algorithm for the description problem.*

It can be easily checked that the constructed formula $\varphi(M)$ satisfies the condition $\mathrm{Sol}(\varphi(M)) = M$. □

The main algorithm that implements the construction of a formula φ in CNF for a given set of vectors M over a finite totally ordered domain D, satisfying the condition $\mathrm{Sol}(\varphi) = M$, and the algorithm computing the fork function are displayed in Figure 3.3.

THEOREM 3.6. *For each set of vectors $M \subseteq D^\ell$ over a finite ordered domain D there exists a formula φ in CNF such that $M = \mathrm{Sol}(\varphi)$. It contains at most $2\,|M|\,\ell$ clauses and its length is $O(|M|\,\ell^2 \log |D|)$. The algorithm constructing φ runs in time*

$O(|M|\,\ell^2 \log|D|)$.

Proof. The formula $\varphi(M)$ contains at most $|M|-1$ middle-clauses and at most $\ell + (|M|-1)(\ell-1)$ left/right-clauses. Summing up these partial bounds, we obtain for the total number of clauses the bound

$$|M| - 1 + 2(\ell + (|M|-1)(\ell-1)) = 2\,|M|\,\ell - |M| + 1 \;\le\; 2\,|M|\,\ell \quad (\text{for } M \neq \emptyset).$$

Each clause contains at most 2ℓ literals, namely at most one positive and one negative for each variable. Each literal has length $O(\log|D|)$, since the domain elements are written in binary notation. Hence, the overall length of the formula $\varphi(M)$ is $O(|M|\,\ell^2 \log|D|)$.

The vectors in M can be lexicographically sorted in time $O(|M|\,\ell \log|D|)$ using a decision tree (trie) or a radix sort. The factor $\log|D|$ stems from the comparison of domain elements. The fork levels can also be computed in time $O(|M|\,\ell \log|D|)$, in parallel with sorting the set M. The formula $\varphi(M)$ is produced by two loops, where the outer loop is going through each vector in M and the inner loop through the variables. The three clauses $\neg\,\mathrm{middle}(k,i)$, $\neg\,\mathrm{left}(k+1,i)$, and $\neg\,\mathrm{right}(k,i)$ are potentially written in each step inside the combined loops. This makes an algorithm with time complexity $O(|M|\,\ell^2 \log|D|)$. ∎

An important property of our algorithm is its linearity with respect to the number of models $|M|$ being the most relevant parameter. In fact, the paper by Amilhastre, Fargier, and Marquis [1] mentions an industrial problem provided by Renault DVI, where the cardinality of the set of vectors is $|M| = 1.5 \cdot 10^{12}$ with the vector arity $\ell = 101$ over a domain of size $|D| = 43$. The complement set $\bar{M}$ contains $43^{101} - 1.5 \cdot 10^{12}$ vectors; therefore, the naive algorithm is inapplicable in this situation.

4. Prime formulas. The formulas in CNF computed by Algorithm DESCRIPTION in section 3 are of a particular form: The variables in each clause form a prefix of the variable vector $(x_1, \ldots, x_\ell)$. As a consequence, although polynomial in the size of the initial relation M, the size of the formulas is not minimal. In Example 3.5, for instance, it can be easily seen that several literals and even clauses can be removed from the formula. We investigate in this section a way to shorten formulas. To this aim we generalize the notion of a *prime* formula in propositional logic to the case of finite domains and show how to obtain such a prime formula in CNF describing the given relation M. Note that although we apply the minimization process to the formulas produced by our algorithm, it can be applied to any formula in CNF.

The notion of primality and prime clauses in many-valued logic was considered for the first time by Murray and Rosenthal in [23].

4.1. Notions of primality. Recall that a clause of a propositional formula φ in CNF is said to be *prime* (with respect to φ) if φ implies none of its proper subclauses. This leads to the following straightforward generalization. Let φ be a CNF formula over some finite totally ordered domain. A clause $c = (l_1 \vee \cdots \vee l_q)$ of φ is said to be *prime* (with respect to φ) if for each $i = 1, \ldots, q$ there exists a model $m_i \in \mathrm{Sol}(\varphi)$ not satisfying the reduced clause $c \setminus l_i = (l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \cdots \vee l_q)$. The formula φ is said to be prime if all its clauses are prime.

However, this notion of primality considers each literal as a whole. This is adequate in the case of classical propositional logic but does not meet our requirements in the case of larger domains. The following more sophisticated notion of primality also considers the value d involved in a literal $x \le d$ or $x \ge d$.

DEFINITION 4.1 (primality). *For a variable x and a pair of values $d, d' \in D$ satisfying the relation $d' > d$ (resp., $d' < d$), the literal $x \ge d'$ (resp., $x \le d'$) is said*

to be stronger *than the literal $x \geq d$ (resp., $x \leq d$). The constant false is stronger than any other nonfalse literal. Removal of a literal from a clause is a particular case of strengthening, namely, of this literal to the constant false.*

Let φ be a CNF formula over a finite totally ordered domain D. A clause c in φ is prime *with respect to φ if strengthening any literal in c yields a clause which is not implied by φ. A formula φ is* prime *if all its clauses are prime.*

As in the propositional case, it is easily seen that for a given formula φ there always exists at least one prime formula φ' which is logically equivalent to φ and can be obtained from φ by strengthening and removing some of its literals. If φ is already prime, then φ and φ' are identical.

The interest in this new, stronger notion of primality comes from efficiency requirements. In fact, the presence of a literal $x \leq d$ or $x \geq d$ instead of $x \leq d'$ or $x \geq d''$, respectively, for $d < d'$ or $d > d''$ reduces the search space during a search for a suitable satisfying assignment.

4.2. Algorithm. Given a CNF formula φ, we show how to efficiently compute a prime formula φ' satisfying the equality $\mathrm{Sol}(\varphi) = \mathrm{Sol}(\varphi')$. Our algorithm, specified in Figure 4.1, is inspired by the one presented in [25] for the Boolean domain. The algorithm considers each clause $l_1 \vee \cdots \vee l_q$ of φ separately. First, it determines for each model m_k the last literal satisfied by it and stores the index of the literal in the array $last[k]$ (lines 3–9). Then the literals are strengthened in turn, starting with the first one.

Suppose that the literal is positive; i.e., it is of the form $x_i \geq d$ (lines 12–20). Strengthening means to increase the value of d. Some models that satisfied the literal before might not satisfy the literal after strengthening. This is a problem only for those models for which the literal was the last possibility to make the clause true (remember that for a model to satisfy a clause it suffices to satisfy a single literal). Therefore, we choose the new value d' as the minimum of all such models (lines 14–19) and construct the new literal as $x_i \geq d'$. Negative literals are handled dually by taking the maximum (lines 21–30). If the literal is redundant, i.e., if no model depends on it as its last literal, the minimum (maximum) would have to be taken over the empty set; in this case we set d' to n (resp., -1).

Line 31 checks whether the literal is redundant. If it is not redundant, it is added to the new clause constructed to replace the old one (line 32). Finally, all models m_k satisfying the new literal are marked by setting $last[k]$ to zero (lines 33–37). As a consequence, the minimum/maximum computations for the remaining literals will ignore these models.

Example 4.2. Let $x_1 \leq 3 \vee x_2 \leq 3 \vee x_3 \leq 0 \vee x_3 \geq 4$ be the clause under consideration, and let M be as in Example 3.5. The array *last* is set to the values

	010	013	220	440	444
last	3	2	3	3	4

The first literal is eliminated since there is no model m_k such that $last[k] = 1$. The second literal is strengthened to $x_2 \leq 1$ since $d' = 1$ for $j = 2$. The last two literals remain unchanged, since the maximum for $j = 3$ is $d' = 0$ and the minimum for $j = 4$ is $d' = 4$. Hence the reduced clause is equal to $x_2 \leq 1 \vee x_3 \leq 0 \vee x_3 \geq 4$.

The PRIMALITY algorithm applied to the whole formula $\varphi(M)$ from Example 3.5

Algorithm: PRIMALITY
Input: A formula φ in conjunctive normal form and a nonempty set $M \subseteq D^\ell$ of vectors such that $\mathrm{Sol}(\varphi) = M$.
Output: A reduced prime formula φ' such that $\mathrm{Sol}(\varphi) = \mathrm{Sol}(\varphi')$.
Method:

```
1: φ' ← true
2: for all clauses c = (l_1 ∨ ··· ∨ l_q) ∈ φ do                ▷ compute the vector last
3:     for k ← 1 to |M| do
4:         for j ← 1 to q do
5:             if m_k satisfies l_j then
6:                 last[k] ← j
7:             end if
8:         end for
9:     end for
10:    c' ← false                                              ▷ greedy strengthening of literals
11:    for j ← 1 to q do
12:        if l_j is positive then
13:            let l_j = x_i ≥ d
14:            d' ← n                  ▷ d' = min({n} ∪ {m_k[i] | 1 ≤ k ≤ |M|, last[k] = j})
15:            for k ← 1 to |M| do
16:                if last[k] = j then
17:                    d' ← min(d', m_k[i])
18:                end if
19:            end for
20:            l' ← x_i ≥ d'
21:        else
22:            let l_j = x_i ≤ d
23:            d' ← −1             ▷ d' = max({−1} ∪ {m_k[i] | 1 ≤ k ≤ |M|, last[k] = j})
24:            for k ← 1 to |M| do
25:                if last[k] = j then
26:                    d' ← max(d', m_k[i])
27:                end if
28:            end for
29:            l' ← x_i ≤ d'
30:        end if
31:        if 0 ≤ d' ≤ n − 1 then
32:            c' ← c' ∨ l'
33:            for k ← 1 to |M| do
34:                if m_k satisfies l' then
35:                    last[k] ← 0
36:                end if
37:            end for
38:        end if
39:    end for
40:    φ' ← φ' ∧ c'
41: end for
42: return φ'
```

FIG. 4.1. *Reduction to a prime formula.*

returns the reduced formula

$$\begin{aligned}\varphi'(M) = {} & (x_3 \le 0 \vee x_3 \ge 3) \wedge (x_1 \le 0 \vee x_1 \ge 2) \wedge (x_1 \le 2 \vee x_1 \ge 4) \\ & \wedge (x_2 \le 1 \vee x_3 \le 0 \vee x_3 \ge 4) \wedge (x_2 \ge 1) \wedge (x_1 \le 0 \vee x_2 \ge 2) \\ & \wedge (x_1 \le 2 \vee x_2 \ge 4) \wedge (x_1 \ge 2 \vee x_2 \le 1) \wedge (x_2 \ge 4 \vee x_3 \le 3) \\ & \wedge (x_1 \ge 4 \vee x_2 \le 2) \wedge (x_2 \le 1 \vee x_2 \ge 4 \vee x_3 \le 0)\,.\end{aligned}$$

Note that $\varphi'(M)$ is a Horn formula with at most three literals per clause. □

THEOREM 4.3. *For a formula φ in conjunctive normal form (CNF) there exists a logically equivalent prime formula φ' such that* $\mathrm{Sol}(\varphi) = \mathrm{Sol}(\varphi')$*, which can be computed in time $O(|\varphi|\,|M|\,\ell \log |D|)$, where $|\varphi|$ is the number of clauses in φ.*

Proof. The time complexity directly follows from Figure 4.1. To prove the correctness of the algorithm, let c be a clause of φ and c' be the clause obtained from c by running Algorithm PRIMALITY.

We first show that the models satisfying c are the same as those satisfying c'. The lines 3–9 set $last[k]$ to a value in $\{1, \ldots, q\}$ for every k, since every model in M satisfies at least one literal in c. Moreover, $last[k]$ is set to zero if and only if the corresponding model satisfies the literal added to c'. Obviously every element of *last* will eventually be set to zero: either the model "accidentally" satisfies a new literal before the last one, or otherwise the new literal derived from the literal identified by $last[k]$ is satisfied by m_k. Therefore, we have $\mathrm{Sol}(\varphi) = \mathrm{Sol}(\varphi')$.

It remains to show that c' is prime. According to Definition 4.1 we have to prove that no literal from c' can be removed or strengthened. Consider the start of the jth iteration of the **for**-loop in lines 11–39. Construct the set $M^{(j)} = \{m_k \mid 1 \le k \le |M|\,, last[k] = j\}$. The models in $M^{(j)}$ satisfy none of the literals added to c' so far (otherwise their *last*-entry would have been set to zero, preventing their inclusion into the set $M^{(j)}$), nor will they satisfy any future literal since the jth literal is the last one satisfied by the models. Therefore, the literal constructed in this iteration cannot be dropped from c' without changing the set of satisfying models. Now suppose that the literal considered in this round is $l_j = x_i \ge d$ (the other case is dual). Let $m^{(j)}$ be one of the models in $M^{(j)}$ for which the ith component is minimal, i.e., $m^{(j)}[i] = d'$. It satisfies $x_i \ge d'$ but clearly no other literal $x_p \ge d_p$ satisfying $d_p > d'$. We conclude that the literals added to c' can be neither removed nor strengthened, which implies that c' is prime. □

Combining Algorithms DESCRIPTION and PRIMALITY, i.e., first describing M by means of a CNF formula $\varphi(M)$, followed by a reduction of $\varphi(M)$ to a prime formula, we get the following result.

COROLLARY 4.4. *For each set of vectors $M \subseteq D^\ell$ over a finite totally ordered domain D there exists a prime formula φ in CNF such that $M = \mathrm{Sol}(\varphi)$. The algorithm constructs φ in time $O(|M|^2\,\ell^2 \log |D|)$.*

5. Horn formulas. Horn clauses and formulas constitute a frequently studied subclass of propositional formulas. This is due to the fact that there exists a polynomial-time algorithm for deciding their satisfiability problem. It turns out that this is still the case for Horn formulas over finite domains [4], which motivates our study of their description and identification problems. As we will see, the sets of vectors described by Horn formulas are closed under the minimum operation.

A question may arise about the usefulness and practical implications of computing a Horn formula $\varphi(M)$ for a given set of vectors M whenever it is possible. Since the complexity of the description algorithm is determined by the cardinality of the

set of vectors M, it may seem superfluous to compute a Horn formula describing them. However, imagine a two-stage procedure, where first a describing formula φ is computed offline for the set of vectors M, followed by its extensive use during a second stage for online reasoning. It is obvious that we prefer a structurally simpler formula φ for the second stage reasoning process. There exist numerous examples in logic and automated deduction (see, e.g., the survey [3] in case of many-valued logics), like resolution or several tableau methods, where it is more efficient to work with Horn clauses or Horn formulas, compared with general formulas in CNF.

Problem: DESCRIPTION[HORN].

Input: A finite set of vectors $M \subseteq D^\ell$, closed under conjunction, over a finite totally ordered domain D.

Output: A Horn formula φ over D such that $\mathrm{Sol}(\varphi) = M$.

The general construction in section 3 does not guarantee that the final formula is Horn whenever the set M is closed under conjunction. For instance, there exists a Horn formula describing the set M presented in Example 2.2, but the formula $\varphi(M)$ computed by the DESCRIPTION algorithm in Example 3.5 is *not* Horn. Therefore, we must reduce the clauses of the formula φ, produced in section 3, to obtain only Horn clauses. For this, we will modify a construction proposed by Jeavons and Cooper in [21]. Their method is exponential, since it proposes to construct a Horn clause for each vector in the complement set $D^\ell \setminus M$. We will first adapt the method of Jeavons and Cooper to get a polynomial-time algorithm and then propose a more sophisticated implementation of the approach that will guarantee us an algorithm with even lower asymptotic complexity.

Let $\varphi(M)$ be a formula produced by the DESCRIPTION algorithm in section 3, and let c be a clause from $\varphi(M)$. We denote by c^- the disjunction of the negative literals in c. The vectors in M satisfying a negative literal in c also satisfy the restricted clause c^-. Hence we have only to care about the vectors that satisfy a positive literal but no negative literals in c, described by the set

$$M_c \;=\; \{m \in M \mid m \not\models c^-\}.$$

If M_c is empty, we can replace the clause c by $h(c) = c^-$ in the formula $\varphi(M)$ without changing the set of models $\mathrm{Sol}(\varphi)$. Otherwise, note that M_c is closed under conjunction, since M is already closed under this operation. Indeed, if the vectors m and m' falsify every negative literal $x \le d$ of c^-, then the conjunction $m \wedge m'$ falsifies the same negative literals. Hence M_c contains a unique minimal model $m_* = \bigwedge M_c$. Every positive literal in c satisfied by m_* is also satisfied by all vectors in M_c. Let l be a positive literal from c and satisfied by m_*. There exists at least one such literal since otherwise m_* would satisfy neither c^- nor any positive literal in c; hence it would not be in M_c. Then c can be replaced with the Horn clause $h(c) = l \vee c^-$, without changing the set of models $\mathrm{Sol}(\varphi)$. We obtain a Horn formula $h(M)$ for a Horn set M by replacing every non-Horn clause c in $\varphi(M)$ by its Horn restriction $h(c)$.

Example 5.1. Consider again the set of vectors $M = \{010, 013, 220, 440, 444\}$ and the non-Horn clause $c = (x_1 \ge 1 \vee x_2 \le 0 \vee x_2 \ge 2 \vee x_3 \le 0 \vee x_3 \ge 3)$ from Example 3.5. We have $c^- = (x_2 \le 0 \vee x_3 \le 0)$; thus any subclause of c containing c^- is already satisfied by the vectors 010, 220, and 440. We need to keep a positive literal from c in order to satisfy the reduced clause also by the vectors $013, 444 \in M$. In other words, we have $M_c = \{013, 444\}$. The unique minimal model is $m_* = 013 \wedge 444 = 013$. Since the minimal model m_* satisfies the positive literal $(x_3 \ge 3)$ in c, we can reduce the clause c to $h(c) = (x_3 \ge 3 \vee x_2 \le 0 \vee x_3 \le 0)$. ∎

The length of $h(M)$ is basically the same as that of $\varphi(M)$. The number of clauses is the same and the length of clauses is $O(\ell \log |D|)$ in both cases. There are at most 2ℓ literals in each clause of $\varphi(M)$ (one positive and one negative literal per variable) versus $\ell + 1$ literals in each clause of $h(M)$ (one negative literal per variable plus a single positive literal).

The construction of each Horn clause $h(c)$ requires time $O(|M|\,\ell \log |D|)$. Indeed, for every vector $m \in M$ we have to evaluate at most ℓ negative literals in c to find out whether m belongs to M_c. The evaluation of a literal takes time $O(\log |D|)$. Hence the computation of the set M_c takes time $O(|M|\,\ell \log |D|)$. To obtain $m_* = \bigwedge M_c$, we have to compute $|M_c|-1$ conjunctions between vectors of length ℓ, each of the ℓ conjunctions taking time $O(\log |D|)$. Therefore, m_* can also be computed in time $O(|M|\,\ell \log |D|)$. Since there are at most $2\,|M|\,\ell$ clauses in $\varphi(M)$, the transformation of $\varphi(M)$ into $h(M)$ can be done in time $O(|M|^2\,\ell^2 \log |D|)$. Hence, the whole algorithm producing the Horn formula $h(M)$ from the set of vectors M runs in time $O(|M|^2\,\ell^2 \log |D|)$.

Note that we can also use the PRIMALITY algorithm to reduce a CNF formula φ to a Horn formula $h(\varphi)$ whenever there exists a Horn formula logically equivalent to φ. The application of the PRIMALITY algorithm yields the same asymptotic time complexity as the aforementioned method according to Corollary 4.4. However, neither the application of the PRIMALITY algorithm nor the aforementioned method are asymptotically optimal.

Another interest for using the primality algorithm comes from the fact that this method does not need the assumption that M is closed under conjunction. Indeed, once we compute a prime formula describing M, we can conclude that M is Horn if the obtained prime formula is Horn. We will return to this issue in section 6 on bijunctive formulas.

We now describe a new algorithm that significantly outperforms the previous methods in terms of running time. This new algorithm is inspired by the one from [17] for the Boolean case. The basic idea is to describe the set M directly by means of a CNF formula as in section 3, but keeping only one or no positive literal in each obtained clause. For this purpose, we define the terms hmiddle(k, i), hleft$(k+1, i)$, and hright(k, i) that replace the corresponding terms defined in section 3. The replacement of the previous terms by the new ones is based on the following lemma.

LEMMA 5.2. *Let $M \subseteq D^\ell$ be a finite set of vectors closed under conjunction and let c be a clause satisfied by each model $m \in M$. Then there exists a Horn subclause $h(c)$ of c that is satisfied by every model from M.*

Proof. If c contains only one or no positive literals, then we set $h(c)$ equal to c. Otherwise, let $c = (x_i \geq a \vee x_j \geq b \vee c')$ be a clause containing two different positive literals, where $i \neq j$ since otherwise one of the two positive literals would be implied by the other and could therefore be eliminated. Assume that neither $x_i \geq a$ nor $x_j \geq b$ can be removed from c if the truth of c with respect to M has to be preserved. Then there must be two models $m, m' \in M$ satisfying the conditions $m[i] \geq a$, $m[j] < b$ and $m'[i] < a$, $m'[j] \geq b$. Moreover, neither m nor m' satisfy the rest of the clause c'. Then it can be easily seen that $(m \wedge m')[i] < a$, $(m \wedge m')[j] < b$, and that the model $m \wedge m'$ does not satisfy c'. Hence, the model $m \wedge m' \in M$ does not satisfy the clause c. This contradicts the hypothesis because M is closed under conjunction. ∎

We now define the terms for the Horn formulas by distinguishing the cases hmiddle, hleft, and hright. The conditions of their application are inherited from

section 3. The first two cases are relatively easy to determine:

$$\text{hmiddle}(k,i) \;=\; \bigwedge_{j<i}(x_j \geq m_k[j]) \;\wedge\; (x_i > m_k[i]) \wedge (x_i < m_{k+1}[i]),$$
$$\text{hleft}(k+1,i) \;=\; \bigwedge_{j<i}(x_j \geq m_{k+1}[j]) \;\wedge\; (x_i < m_{k+1}[i]).$$

We can easily see that $\neg\,\text{hmiddle}(k,i)$ and $\neg\,\text{hleft}(k+1,i)$ are Horn clauses. Observe that $\neg\,\text{hleft}(k+1,i)$ implies $\neg\,\text{left}(k+1,i)$. We will show that the rightmost literal cannot be removed from $\neg\,\text{left}(k+1,i)$ when we compute the Horn subclause $\neg\,\text{hleft}(k+1,i)$. From the tree T_M and the term $\text{left}(k+1,i)$ observe that the clause $c = \neg\,\text{left}(k+1,i) \setminus (x_i \geq m_{k+1}[i])$ is falsified by at least one model in M. Therefore, there is no Horn subclause $h(c)$ of c that is satisfied by all models in M. Since M is closed under conjunction, from Lemma 5.2 it follows that there must be a Horn subclause $h(c)$ of c that is satisfied by all models of M. Hence, the clause $h(c)$ must contain the literal $x_i \geq m_{k+1}[i]$ and since it is Horn, it must be a subclause of $\neg\,\text{hleft}(k+1,i)$. Similarly for $\text{hmiddle}(k,i)$, the construction ensures that the rightmost literal cannot be removed from $\neg\,\text{middle}(k,i)$, since otherwise the clause $\neg\,\text{middle}(k,i)$ would be equal to $\neg\,\text{right}(k,i)$. Thus all other positive literals can be removed from $\neg\,\text{middle}(k,i)$ to obtain the Horn subclause $\neg\,\text{hmiddle}(k,i)$.

The construction of the term $\text{hright}(k,i)$ is more involved. Recall that for all convenient parameters k and i we have the clause

$$\neg\,\text{right}(k,i) \;=\; \bigvee_{j<i}(x_j < m_k[j] \;\vee\; x_j > m_k[j]) \;\vee\; (x_i \leq m_k[i]).$$

We look for the positive literals that can be removed from $\neg\,\text{right}(k,i)$ in order to derive the term $\text{hright}(k,i)$. For this purpose, construct the set

$$M(k,i) \;=\; \{m \in M \mid m[i] > m_k[i] \text{ and } \forall j < i,\; m[j] \geq m_k[j]\}$$

for the given parameters k and i. Clearly, $M(k,i)$ is the set of all vectors from M that falsify all negative literals in $\neg\,\text{right}(k,i)$. The set $M(k,i)$ corresponds to the previously defined set M_c for $c = \neg\,\text{right}(k,i)$.

We distinguish the cases $M(k,i) = \emptyset$ and $M(k,i) \neq \emptyset$. When the set $M(k,i)$ is empty, this means that every model $m \in M$ satisfies at least one negative literal in $\neg\,\text{right}(k,i)$. Thus we can construct $\neg\,\text{hright}(k,i)$ from $\neg\,\text{right}(k,i)$ by removing all positive literals. In other words, $\text{hright}(k,i)$ is obtained from $\text{right}(k,i)$ by removing all negative literals.

When the set $M(k,i)$ is nonempty, we know from Lemma 5.2 that there exists a positive literal in $\neg\,\text{right}(k,i)$ which is satisfied by all models in $M(k,i)$. This amounts to the computation of intersection $\bigwedge M(k,i)$ followed by a choice of a model from it, but we need to do it in a more sophisticated way than in the previous Horn method if we wish to obtain an algorithm with lower asymptotic complexity. Let us define a function that will compute the position of the kept literal:

$$\text{pos}(k,i) \;=\; \max_{1\leq j\leq k}\{j \mid \exists m \in M(k,i) \text{ such that } \forall p,\; p < j \text{ implies } m[p] \leq m_k[p]\}.$$

Note that since every model $m \in M(k,i)$ satisfies the clause $\neg\,\text{right}(k,i)$ and thus at least one of its positive literals, the condition $\text{pos}(k,i) < i$ is satisfied.

Algorithm: POS
Input: Nonempty set $M \subseteq D^\ell$ of vectors and a parameter k.
Output: The position function $pos(k)$.
Method:

```
 1: for i ← 1 to ℓ do
 2:     pos(k)[i] ← 0
 3: end for
 4: for k' ← 1 to |M| do
 5:     if k' ≠ k then
 6:         i_0 ← 1
 7:         while i_0 < ℓ and m_{k'}[i_0] ≤ m_k[i_0] do
 8:             i_0 ← i_0 + 1
 9:         end while
10:         i ← 1
11:         while i ≤ ℓ and m_{k'}[i] ≥ m_k[i] do
12:             if i > fork(k) and m_k[i] < |D| − 1 and m_{k'}[i] > m_k[i] then
13:                 pos(k)[i] ← max(pos(k)[i], i_0)
14:             end if
15:             i ← i + 1
16:         end while
17:     end if
18: end for
19: return pos(k)
```

FIG. 5.1. *Position function* $\mathrm{pos}(k,i)$.

The term $\mathrm{hright}(k,i)$ is defined as follows:

$$\mathrm{hright}(k,i) = \begin{cases} \bigwedge_{j<i}(x_j \geq m_k[j]) \ \wedge \ (x_i > m_k[i]) & \text{if } M(k,i) = \emptyset, \\ \bigwedge_{j<i}(x_j \geq m_k[j]) \ \wedge \ (x_i > m_k[i]) \ \wedge \ (x_{\mathrm{pos}(k,i)} \leq m_k[\mathrm{pos}(k,i)]) & \\ \qquad \text{otherwise.} & \end{cases}$$

We can finally present the Horn formula $h(M)$ for a set of vectors M closed under conjunction that satisfies the equality $\mathrm{Sol}(h(M)) = M$. It resembles the formula $\varphi(M)$ from section 3 modulo some syntactic changes.

$$\begin{aligned} h(M) \quad &= \quad \bigwedge\{ \neg\,\mathrm{hmiddle}(k,i) \mid 0 < k < |M|\,,\ i = \mathrm{fork}(k),\ m_k[i]+1 < m_{k+1}[i] \} \\ &\wedge \quad \bigwedge\{ \neg\,\mathrm{hleft}(k+1,i) \mid 0 \leq k < |M|\,,\ \mathrm{fork}(k) < i \leq \ell,\ m_{k+1}[i] > 0 \} \\ &\wedge \quad \bigwedge\{ \neg\,\mathrm{hright}(k,i) \quad \mid 0 < k \leq |M|\,,\ \mathrm{fork}(k) < i \leq \ell,\ m_k[i] < |D|-1 \}. \end{aligned}$$

Our algorithm computing the Horn formula $h(M)$ for a given set of vectors M is equivalent to Algorithm DESCRIPTION, where the terms $\mathrm{middle}(k,i)$, $\mathrm{left}(k+1,i)$, and $\mathrm{right}(k,i)$ are replaced by the terms $\mathrm{hmiddle}(k,i)$, $\mathrm{hleft}(k+1,i)$, and $\mathrm{hright}(k,i)$, respectively. Computing the clauses $\neg\,\mathrm{hmiddle}(k,i)$ and $\neg\,\mathrm{hleft}(k+1,i)$ does not pose any problems, whereas the clause $\neg\,\mathrm{hright}(k,i)$ heavily depends on the position function $\mathrm{pos}(k,i)$ which must be computed efficiently. For a given k, Algorithm POS in Figure 5.1 computes the values of $\mathrm{pos}(k,i)$ for all positions $i = 1,\ldots,\ell$ at the same

time. This is what makes our algorithm efficient. The result is returned in the form of an array $pos(k)$, where the equality $pos(k)[i] = \mathrm{pos}(k, i)$ holds for all positions i.

The algorithm is based on the following principle. Given a vector m_k, it considers every vector $m_{k'} \in M$ different from m_k. For a pair of vectors m_k and $m_{k'}$, the algorithm considers each index i in increasing order. While the condition $m_{k'}[i] \geq m_k[i]$ (line 11) holds, we are sure that the model $m_{k'}$ belongs to $M(k, i)$, and therefore $m_{k'}$ is taken into account for the value of $\mathrm{pos}(k, i)$ (line 13). On the other hand, as soon as the condition does not hold any more, then we are sure that for all $j > i$ the vector $m_{k'}$ does not belong to $M(k, i)$, and therefore $m_{k'}$ does not need to be taken into account for the value of $\mathrm{pos}(k, i)$. Note that the value $pos[k, i]$ has no meaning for a nonconvenient i, and therefore it is set to 0, which also means $M(k, i) = \emptyset$.

Example 5.3. We already know from Example 4.2 that there exists a Horn formula describing the set of vectors $M = \{010, 013, 220, 440, 444\}$. Let us transform the terms middle, left, and right from Example 3.3 to hmiddle, hleft, and hright, respectively. We will get the middle terms

$$\begin{aligned}
\mathrm{hmiddle}(1,3) &= (x_1 \geq 0) \wedge (x_2 \geq 1) \wedge (x_3 > 0) \wedge (x_3 < 3),\\
\mathrm{hmiddle}(2,1) &= (x_1 > 0) \wedge (x_1 < 2),\\
\mathrm{hmiddle}(3,1) &= (x_1 > 2) \wedge (x_1 < 4),\\
\mathrm{hmiddle}(4,3) &= (x_1 \geq 4) \wedge (x_2 \geq 4) \wedge (x_3 > 0) \wedge (x_3 < 4)
\end{aligned}$$

and the left terms

$$\begin{aligned}
\mathrm{hleft}(1,2) &= (x_1 \geq 0) \wedge (x_2 < 1),\\
\mathrm{hleft}(3,2) &= (x_1 \geq 2) \wedge (x_2 < 2),\\
\mathrm{hleft}(4,2) &= (x_1 \geq 4) \wedge (x_2 < 4)
\end{aligned}$$

easily. To transform the terms $\mathrm{right}(2,2)$, $\mathrm{right}(3,2)$, and $\mathrm{right}(3,3)$ to $\mathrm{hright}(2,2)$, $\mathrm{hright}(3,2)$, and $\mathrm{hright}(3,3)$, respectively, we need first to compute the arrays $pos(2)$ and $pos(3)$. We get

pos	1	2	3
2	0	1	1
3	0	1	1

which implies the terms

$$\begin{aligned}
\mathrm{hright}(2,2) &= (x_1 \geq 0) \wedge (x_2 > 1) \wedge (x_1 \leq 0),\\
\mathrm{hright}(2,3) &= (x_1 \geq 0) \wedge (x_2 \geq 1) \wedge (x_3 > 3) \wedge (x_1 \leq 0),\\
\mathrm{hright}(3,2) &= (x_1 \geq 2) \wedge (x_2 > 2) \wedge (x_1 \leq 2),\\
\mathrm{hright}(3,3) &= (x_1 \geq 2) \wedge (x_2 \geq 2) \wedge (x_3 > 0) \wedge (x_1 \leq 2).
\end{aligned}$$

The final Horn formula will be

$$\begin{aligned}
h(M) &= (x_2 \leq 0 \vee x_3 \leq 0 \vee x_3 \geq 3) \wedge (x_1 \leq 0 \vee x_1 \geq 2) \wedge (x_1 \leq 2 \vee x_1 \geq 4)\\
&\wedge\ (x_1 \leq 3 \vee x_2 \leq 3 \vee x_3 \leq 0 \vee x_3 \geq 4) \wedge (x_2 \geq 1) \wedge (x_1 \leq 1 \vee x_2 \geq 2)\\
&\wedge\ (x_1 \leq 3 \vee x_2 \geq 4) \wedge (x_1 \geq 1 \vee x_2 \leq 1) \wedge (x_1 \geq 1 \vee x_2 \leq 0 \vee x_3 \leq 3)\\
&\wedge\ (x_1 \leq 1 \vee x_1 \geq 3 \vee x_2 \leq 2) \wedge (x_1 \leq 1 \vee x_1 \geq 3 \vee x_2 \leq 1 \vee x_3 \leq 0). \qquad \square
\end{aligned}$$

THEOREM 5.4. *For each set of vectors $M \subseteq D^\ell$ over a finite totally ordered domain D that is closed under conjunction, there exists a Horn formula φ such that $M = \mathrm{Sol}(\varphi)$. The formula φ contains at most $2\,|M|\,\ell$ clauses, its length is $O(|M|\,\ell^2 \log |D|)$, and it can be computed in time $O(|M|\,\ell(|M| + \ell) \log |D|)$.*

Proof. Time complexity is straightforward since we essentially run the DESCRIPTION algorithm from section 3 with the computation of $pos(k)$ added. As for correctness, it is a consequence of the previously written reasoning in this section. ∎

Using Theorem 5.4, we are able to prove a generalization of a well-known characterization of the set of models $\mathrm{Sol}(\varphi)$ of a Horn formula φ. We wish to point out that this characterization is not new and was proved before. Indeed, a related characterization in a different setting can be found in [15], and a similar proof can be found in [21]. We mention the result here for completeness.

PROPOSITION 5.5. *A set of vectors M over a finite totally ordered domain is closed under conjunction if and only if there exists a Horn formula φ satisfying the identity $\mathrm{Sol}(\varphi) = M$.*

Proof. Theorem 5.4 shows that there exists a Horn formula φ describing a set of vectors M if M is closed under conjunction. It remains to show the converse, namely, that the set $\mathrm{Sol}(\varphi)$ is closed under conjunction if φ is a Horn formula. We need to show that for any two models m and m' of φ, their conjunction $m \wedge m'$ is also a model of φ. A model satisfies a Horn formula φ if and only if it satisfies every clause of φ. Therefore, we need to show for each clause c that $m \wedge m'$ satisfies c whenever both m and m' satisfy c. We distinguish two cases.

(i) Clause c contains a positive literal $x \geq d$ that is satisfied by both m and m'. Then we have $m(x) \geq d$ and $m'(x) \geq d$, which implies $(m \wedge m')(x) = \min(m(x), m'(x)) \geq d$. This proves that $m \wedge m'$ satisfies the clause c.

(ii) Clause c contains no positive literal satisfied by both m and m'. Then at least one model, say, m, must satisfy a negative literal $x \leq d$ in c, i.e., $m(x) \leq d$. We obtain $(m \wedge m')(x) = \min(m(x), m'(x)) \leq m(x) \leq d$. Hence also $m \wedge m'$ satisfies c. ∎

If we interchange conjunctions with disjunctions of models, as well as positive and negative literals throughout section 5, we obtain identical results for dual Horn formulas.

THEOREM 5.6. *A set of vectors $M \subseteq D^\ell$ over a finite ordered domain D is closed under disjunction if and only if there exists a dual Horn formula φ satisfying the identity $M = \mathrm{Sol}(\varphi)$. Given M closed under disjunction, the dual Horn formula φ contains at most $2\,|M|\,\ell$ clauses, and its length is $O(|M|\,\ell^2 \log |D|)$. It can be constructed in time $O(|M|\,\ell(|M| + \ell) \log |D|)$.*

6. Bijunctive formulas. Bijunctive clauses and formulas present another frequently studied subclass of propositional formulas, once more because there exists a polynomial-time algorithm for deciding their satisfiability which generalizes to the finite-domain case [5]. We investigate in this section the description problem for a generalization of bijunctive formulas to ordered finite domains, namely, for sets of vectors closed under the median operation.

Problem: DESCRIPTION[BIJUNCTIVE].
Input: A finite set of vectors $M \subseteq D^\ell$, closed under median, over a finite totally ordered domain D.
Output: A bijunctive formula φ over D such that $\mathrm{Sol}(\varphi) = M$.

Once again, the general construction in section 3 does not guarantee that the final formula is bijunctive whenever the set M is closed under median. Therefore,

we add a postprocessing step that transforms the formula φ into a bijunctive one $b(\varphi)$. Let $\varphi(M)$ be the formula produced by the method of section 3, and let c be a clause from $\varphi(M)$. We construct a bijunctive restriction $b(\varphi)$ by removing appropriate literals from φ such that no more than two literals remain in each clause. Since φ is a CNF, any model of $b(\varphi)$ is still a model of φ. The converse does not hold in general. However, if $\mathrm{Sol}(\varphi)$ is closed under median, the method presented below preserves the models; i.e., every model of φ remains a model of $b(\varphi)$. In the proof we need the following simple lemma.

LEMMA 6.1. *The model* $\mathrm{med}(m_1, m_2, m_3)$ *satisfies a literal* l *if and only if at least two of the models* m_1*,* m_2*, and* m_3 *satisfy* l*.*

Proof. Recall the identities $\mathrm{med}(m_1, m_2, m_3) = (m_1 \vee m_2) \wedge (m_2 \vee m_3) \wedge (m_3 \vee m_1) = (m_1 \wedge m_2) \vee (m_2 \wedge m_3) \vee (m_3 \wedge m_1)$. Recall also that $m \wedge m'$ and $m \vee m'$ are shorthand for the more cumbersome prefix notation $\min(m, m')$ and $\max(m, m')$, respectively. Let l be satisfied by at least two models, say, m_1 and m_2. If the literal l is positive, then it is also satisfied by the models $m_1 \vee m_2$, $m_2 \vee m_3$, and $m_3 \vee m_1$. If the literal l is negative, then it is also satisfied by the models $m_1 \wedge m_2$, $m_2 \wedge m_3$, and $m_3 \wedge m_1$. Hence, in both cases, l is also satisfied by $\mathrm{med}(m_1, m_2, m_3)$.

Conversely, if l is satisfied only by one model, say, m_1, or if l is not satisfied by any of the three models, then it is falsified by the model $m_2 \vee m_3$ if l is positive and by the model $m_2 \wedge m_3$ if l is negative. Hence, the literal l cannot be satisfied by $\mathrm{med}(m_1, m_2, m_3)$. ∎

DEFINITION 6.2. *We say that a literal* l *is* essential *for a clause* c *with respect to a set of models* M *if there is a model* $m \in M$ *that satisfies* l*, but no other literal in* c*. We also say that* m *is a* justification *for* l *with respect to* M*.*

Obviously, we may remove nonessential literals from c without losing models. It remains to show that no clause from φ contains more than two essential literals.

To derive a contradiction, suppose that c is a clause from φ containing at least three essential literals, say, l_1, l_2, and l_3. Let m_1, m_2, and m_3 be their justifications; i.e., for each i we have $m_i \models l_i$ and m_i does not satisfy any other literal in c. According to Lemma 6.1, in this case the model $\mathrm{med}(m_1, m_2, m_3)$ satisfies no literal at all. Hence $\mathrm{med}(m_1, m_2, m_3)$ satisfies neither c nor φ, which contradicts the assumption that $\mathrm{Sol}(\varphi)$ is closed under median.

The previous discussion suggests applying the following algorithm to every clause c of φ. For every literal l in $c = c' \vee l$, check whether the remaining clause c' is still satisfied by all models in M. If the answer is yes, the literal is not essential and can be removed. Otherwise, it is one of the (at most) two literals in the final bijunctive clause $b(c)$. As we can easily see, this operation is performed by the PRIMALITY algorithm from section 4.

THEOREM 6.3. *For each set of vectors* $M \subseteq D^\ell$ *over a finite ordered domain* D *that is closed under median, there exists a bijunctive formula* φ *such that* $M = \mathrm{Sol}(\varphi)$*. The length of* φ *is* $O(|M|\,\ell(\log \ell + \log |D|))$*, and it contains at most* $2\,|M|\,\ell$ *clauses. The algorithm constructing* φ *runs in time* $O(|M|^2\,\ell^2 \log |D|)$*.*

Proof. The main part of the result follows from Theorems 3.6 and 4.3. Concerning the length of φ, note that each clause contains at most two literals. Each literal consists of a variable and one domain element. Hence we can represent a literal by the index of its variable and the domain value, both written in binary. Therefore, the length of a literal and subsequently of each bijunctive clause is $O(\log \ell + \log |D|)$. Since there are at most $2\,|M|\,\ell$ clauses, this implies the length of φ. ∎

Similarly to the Horn and dual Horn formulas, we get a nice relation between bijunctive formulas by means of a closure property.

PROPOSITION 6.4. *A set of vectors M over a finite totally ordered domain is closed under median if and only if there exists a bijunctive formula φ satisfying the identity $M = \mathrm{Sol}(\varphi)$.*

Proof. Theorem 6.3 shows that there exists a bijunctive formula for every set of vectors M closed under median. It remains to show the converse, namely, that $\mathrm{Sol}(\varphi)$ is closed under median if φ is a bijunctive formula. Since φ is a conjunction of clauses, it is sufficient to show the closure property for a bijunctive clause $c = l \vee l'$. Let m_1, m_2, and m_3 be three models of c. From the pigeonhole principle it follows that one of the two literals l or l' of the clause c is satisfied by at least two models. Hence, by Lemma 6.1, at least one of the two literals l and l', and therefore also the clause c, is satisfied by $\mathrm{med}(m_1, m_2, m_3)$. ∎

We wish to point out that contrary to the Horn case, the most efficient known algorithm for the bijunctive description problem does not seem to lift well from the Boolean to the finite domain. Dechter and Pearl [11] showed that in the Boolean case this problem can be solved in time $O(|M|\,\ell^2)$, which is better than our result even when ignoring the unavoidable factor $\log|D|$. Their algorithm generates first all the $O(\ell^2)$ bijunctive clauses built from the variables of the formula, followed by an elimination of those falsified by a vector from M, where the bijunctive formula is the conjunction of the retained clauses. However, there are $O(\ell^2\,|D|^2)$ bijunctive clauses for a finite domain D yielding an algorithm with time complexity $O(|M|\,\ell^2\,|D|^2)$, which is exponential in the size $O(\log|D|)$ of the domain elements.

Another idea, not applicable more efficiently in the finite domain case, is that of projecting M onto each pair of variables and then computing a bijunctive formula for each projection. This requires time $O(|M|\,\ell^2)$ in the Boolean case, since we need only to compute a CNF for each projection. A CNF for a projection is always bijunctive; thus only the general Algorithm DESCRIPTION has to be used. However, in the finite domain case, computing a formula with Algorithm DESCRIPTION does not necessarily yield a bijunctive CNF. Each clause can contain up to four literals, a positive and a negative one for each variable. Thus we need to use an algorithm for computing a *bijunctive* CNF, like that of Theorem 6.3, yielding an overall time complexity of $O(|M|^2\,\ell^2 \log|D|)$.

Finally, let us return to the identification problem. As was mentioned in the introduction, our results present an elegant and unified algorithm for identification of Horn, dual Horn, and bijunctive sets of vectors. Given a set of vectors M, the method presented by the PRIMALITY algorithm computes a prime formula φ satisfying the identity $\mathrm{Sol}(\varphi) = M$. Then we check in linear time whether it is Horn, dual Horn, or bijunctive. The results in sections 5 and 6 ensure that the resulting formula φ is Horn, dual Horn, or bijunctive if and only if M is a Horn, dual Horn, or bijunctive set of vectors, respectively. This generalizes the result presented by Zanuttini and Hébrard in [25].

7. Changing the literals. We finish the paper with a short discussion of the description problems for a slightly different formalism, where only the literals are changed.

If we change the underlying notion of literals, using the expressions $x = d$ and $x \neq d$ as basic building blocks, the situation changes drastically. Former positive literal $x \geq d$ becomes shorthand for the disjunction $(x = d) \vee (x = d+1) \vee \cdots \vee (x = n-1)$, whereas the former negative literal $x \leq d$ now represents the disjunction $(x = 0) \vee (x = 1) \vee \cdots \vee (x = d)$. Even if we compress literals containing the same variable into a bit vector, the new representation still needs n bits; i.e., its size is $O(n)$.

Compared to the former literals of size $O(\log n)$, this amounts to an exponential blow-up. As an immediate consequence, the algorithms given in the preceding sections become exponential, since we have to replace literals like $x_i < m_k[i]$, $x_i > m_k[i]$, and $x_i < m_{k+1}[i]$ by disjunctions of equalities.

The satisfiability problem for formulas in CNF over finite totally ordered domains with basic operators $\leq$ and $\geq$ is defined similarly to Boolean satisfiability. The complexity of these problems was studied for fixed domain cardinalities, from the standpoint of many-valued logics, by Béjar, Hähnle, and Manyà [6] and Hähnle [16]. The NP-completeness proof for Boolean satisfiability generalizes uniformly to finite ordered domains. Béjar, Hähnle, and Manyà [6] and Hähnle [16] proved that the satisfiability problems restricted to Horn, dual Horn, and bijunctive formulas are decidable in polynomial time for a fixed domain cardinality. These algorithms can be generalized to arbitrary domain cardinalities, adding only the unavoidable factor $\log |D|$.

The satisfiability of formulas in CNF is also affected when switching to $=$ and $\neq$ as basic operators. While the satisfiability problem for general formulas remains NP-complete, the restrictions to Horn, dual Horn, and bijunctive formulas change from polynomially solvable to NP-complete for $|D| \geq 3$. This can be shown by encoding, for example, the graph problem of k-COLORING [2, 9]. When we use the Horn and bijunctive clause $(u \neq d \vee v \neq d)$, we can express by $C(u, v) = (u \neq 0 \vee v \neq 0) \wedge \cdots \wedge (u \neq k-1 \vee v \neq k-1)$ that the adjacent vertices of the edge (u, v) are "colored" by different "colors." On the other hand, Beckert, Hähnle, and Manyà [5] proved that bijunctive formulas restricted to positive literals can be solved in linear time.

8. Concluding remarks. The studied formula description problems constitute a generalization of the Boolean structure identification problems, studied by Dechter and Pearl [11], with more efficient algorithms as a byproduct. Our paper presents a complement to the work of Hähnle et al. [5, 6, 16] on the complexity of the satisfiability problems in many-valued logics. It also completes the study of tractable formulas [9, 20, 21] by Jeavons and his group.

We have constructed an efficient polynomial-time algorithm for the formula description problem over a finite totally ordered domain, where the produced formula is in CNF. We have then presented a subsequent algorithm that eliminates in polynomial time redundancies from the previously computed formula, given the original set of vectors, producing in this way the prime formula. The notion of primality that we use is an extension of the same notion used in Boolean formulas. It not only captures irrelevant literals in clauses but also strengthens the value d in the literals $x \leq d$ or $x \geq d$, respectively. If the original set of vectors is closed under the operation of conjunction, disjunction, or median, we have presented specific algorithms that produce a Horn, a dual Horn, or a bijunctive formula, respectively. These algorithms generalize the well-known ones from the Boolean domain. It is interesting to note that they are compatible, with respect to asymptotic complexity, with known algorithms for the Boolean case presented in [17, 25]. This means that the restriction of the new algorithms presented in our paper to domains D with cardinality $|D| = 2$ produces the aforementioned algorithms for the Boolean case. We also found that in the case when the finite domain is totally ordered, the three well-known special cases, namely, Horn, dual Horn, and bijunctive, display the same behavior as in the Boolean case: (1) they have polynomial satisfiability algorithms and (2) they have the same closure properties.

It would be interesting to know if more efficient algorithms exist or whether

our algorithms are asymptotically optimal. Certainly a more involved lower bound analysis is necessary to answer this open question. Another possible extension of our work would be a generalization of our algorithms to partially ordered domains and to domains with a different structure, like lattices. An additional direction for future work is to look at infinite domains. Some related complexity results for satisfiability problems in the infinite domain case can be found in [4].

REFERENCES

[1] J. Amilhastre, H. Fargier, and P. Marquis, *Consistency restoration and explanations in dynamic CSPs—application to configuration*, Artificial Intelligence, 135 (2002), pp. 199–234.

[2] C. Ansótegui and F. Manyà, *New logical and complexity results for signed-SAT*, in Proceedings of the 33rd IEEE International Symposium on Multiple-Valued Logic (ISMVL 2003), Tokyo, Japan, 2003, IEEE Computer Society, Washington, DC, 2003, pp. 181–187.

[3] M. Baaz, C. G. Fermüller, and G. Salzer, *Automated deduction for many-valued logics*, in Handbook of Automated Reasoning, Vol. 2, J. A. Robinson and A. Voronkov, eds., Elsevier Science, New York, 2001, pp. 1355–1402.

[4] B. Beckert, R. Hähnle, and F. Manyà, *Transformations between signed and classical clause logic*, in Proceedings of the 29th IEEE International Symposium on Multiple-Valued Logic (ISMVL 1999), Freiburg im Breisgau, Germany, 1999, IEEE Computer Society, Washington, DC, 1999, pp. 248–255.

[5] B. Beckert, R. Hähnle, and F. Manyà, *The 2-SAT problem of regular signed CNF formulas*, in Proceedings of the 30th IEEE International Symposium on Multiple-Valued Logic (ISMVL 2000), Portland, OR, 2000, IEEE Computer Society, Washington, DC, 2000, pp. 331–336.

[6] R. Béjar, R. Hähnle, and F. Manyà, *A modular reduction of regular logic to classical logic*, in Proceedings of the 31st IEEE International Symposium on Multiple-Valued Logic (ISMVL 2001), Warsaw, Poland, 2001, IEEE Computer Society, Washington, DC, 2001, pp. 221–226.

[7] A. A. Bulatov, *A dichotomy theorem for constraints on a three-element set*, in Proceedings of the 43rd ACM Symposium on Foundations of Computer Science (FOCS 2002), Vancouver, British Columbia, Canada, 2002, pp. 649–658.

[8] A. A. Bulatov, *Tractable conservative constraint satisfaction problems*, in Proceedings of the 18th IEEE Symposium on Logic in Computer Science (LICS 2003), Ottawa, Canada, 2003, pp. 321–330.

[9] M. C. Cooper, D. A. Cohen, and P. Jeavons, *Characterising tractable constraints*, Artificial Intelligence, 65 (1994), pp. 347–361.

[10] N. Creignou, S. Khanna, and M. Sudan, *Complexity Classifications of Boolean Constraint Satisfaction Problems*, SIAM Monographs on Discrete Mathematics and Applications 7, SIAM, Philadelphia, 2001.

[11] R. Dechter and J. Pearl, *Structure identification in relational data*, Artificial Intelligence, 58 (1992), pp. 237–270.

[12] T. Feder and M. Y. Vardi, *The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory*, SIAM J. Comput., 28 (1998), pp. 57–104.

[13] A. Gil, M. Hermann, G. Salzer, and B. Zanuttini, *Efficient algorithms for constraint description problems over finite totally ordered domains*, in Proceedings of the 2nd International Joint Conference on Automated Reasoning (IJCAR'04), Cork, Ireland, 2004, Lecture Notes in Comput. Sci. 3097, D. Basin and M. Rusinowitch, eds., Springer-Verlag, New York, 2004, pp. 244–258.

[14] R. Hähnle, *Short conjunctive normal forms in finitely valued logics*, J. Logic Comput., 4 (1994), pp. 905–927.

[15] R. Hähnle, *Exploiting data dependencies in many-valued logics*, J. Appl. Non-Classical Logics, 6 (1996), pp. 49–69.

[16] R. Hähnle, *Complexity of many-valued logics*, in Proceedings of the 31st IEEE International Symposium on Multiple-Valued Logic (ISMVL 2001), Warsaw, Poland, 2001, IEEE Computer Society, Washington, DC, 2001, pp. 137–148.

[17] J.-J. Hébrard and B. Zanuttini, *An efficient algorithm for Horn description*, Inform. Process. Lett., 88 (2003), pp. 177–182.

[18] P. Hell and J. Nešetřil, *Graphs and Homomorphisms*, Oxford University Press, Oxford, UK, 2004.
[19] P. Jeavons, *On the algebraic structure of combinatorial problems*, Theoret. Comput. Sci., 200 (1998), pp. 185–204.
[20] P. Jeavons, D. Cohen, and M. Gyssens, *Closure properties of constraints*, J. ACM, 44 (1997), pp. 527–548.
[21] P. Jeavons and M. C. Cooper, *Tractable constraints on ordered domains*, Artificial Intelligence, 79 (1995), pp. 327–339.
[22] P. G. Kolaitis and M. Y. Vardi, *Conjunctive-query containment and constraint satisfaction*, J. Comput. System Sci., 61 (2000), pp. 302–332.
[23] N. V. Murray and E. Rosenthal, *Adapting classical inference techniques to multiple-valued logics using signed formulas*, Fund. Inform., 21 (1994), pp. 237–253.
[24] T. J. Schaefer, *The complexity of satisfiability problems*, in Proceedings of the 10th ACM Symposium on Theory of Computing (STOC'78), San Diego, CA, 1978, pp. 216–226.
[25] B. Zanuttini and J.-J. Hébrard, *A unified framework for structure identification*, Inform. Process. Lett., 81 (2002), pp. 335–339.